Offline Browser

Need

Often we visit sites and find a page of information important. Internet explorer provides a way to download this page for later viewing. But if too many pages are found to be important and useful or if say the complete site is of interest then visiting these many pages and saving them online would be very expensive. What is required is a way to save all the required pages of a site and later view them offline. This would be very cheap.

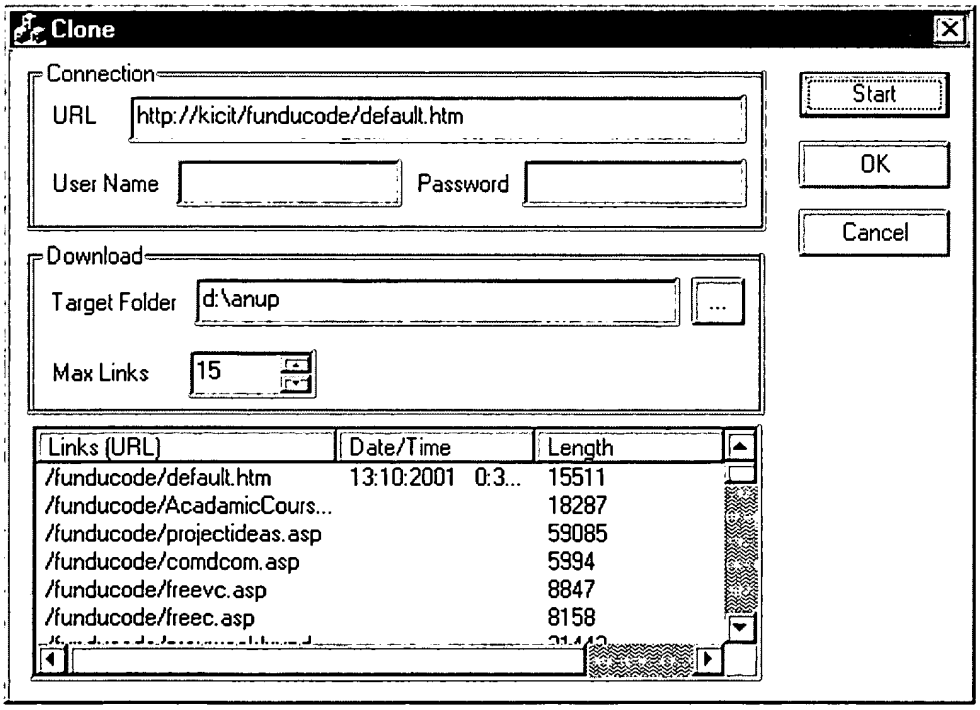
Overview

The software starts off with an http link provided by us. This page is downloaded. Next, all the links in this page are found out and are downloaded too. The newly downloaded pages are again searched for further links. This continues till the maximum link specified by the user have been downloaded or till the complete site is downloaded. While downloading links any duplicate links if found are ignored. Also links to the current page with bookmarks are ignored. There is an option for specifying the target folder for these downloaded files. The application uses a worker-thread to download the files. After a link is downloaded its details: size, status link name etc. are displayed in a list control.

Requirement

- VC++ Platform - for development.
- Internet Explorer 4.0 or above - for necessary DHTML components.
- Personal web Server (PWS) on Windows 98 or Internet Information Server (IIS) on WinNT - for offline testing.

Preview



Steps To Develop This Application

1. Create a Dialog-based application and check ActiveX controls check box in Step 2 of AppWizard.
2. Add the ActiveX control to the project by selecting ' Project | Add to Project | Component and Controls | Registered ActiveX controls' and then double click on 'Microsoft Web Browser' ActiveX control.
3. Insert a **'Microsoft Web Browser'** ActiveX control on the dialog surface by right clicking on the dialog

and selecting **'Insert ActiveX control'**. This would generate a wrapper class named **CWebBrowser2**.

4. Add other controls to the dialog template as shown in the preview.
5. Add member variables for these controls using ClassWizard.
6. In the **OnInitDialog( )** handler, insert columns in the list control using **CListCtrl::InsertColumn( )** & initialize the spin control's range using **CSpinButtonCtrl::SetRange32( )** to a suitable value, say 1000.
7. Add handler for the browse button labeled '...' and use the **SHBrowseForFolder( )** to display a standard folder dialog.
8. Add a handler for the 'Start' button. Inside this handler retrieve the data from various controls to their respective variables. Next use **AfxParseURLEx( )** to separate out various elements from the typed URL such as service type, server name, object name, etc. After this use the 'Microsoft Web Browser ActiveX' control to navigate to the typed URL by calling **CWebBrowser2::Navigate( )** method.
9. Right click on the ActiveX control on the dialog and select events from the context menu. Add the **OnDocumentComplete( )** event handler. In this handler retrieve the **IDispatch** pointer of the Active Document currently contained in the ActiveX control by calling **CWebBrowser2::GetDocument( )**. Using this **IDispatch** pointer obtain **IHTMLDocument2** interface pointer. Now using this pointer call **get_images( )** & **get_links( )** to obtain **IHTMLElementCollection** interface pointers for images representing <img> tags and links representing <a href = > tags respectively. Using **IHtmlElementCollection::get_length( )** acquire the number of items in the collection. Loop through each of the element and using **IHtmlElementCollection::item( )** retrieve a pointer to **IDispatch**. Using this **IDispatch** pointer query for **IHTMLImgElement & IHTMLAnchorElement** once for images and links respectively. In both cases use **get_href( )** method to get the link. In case of links verify whether the link is a valid link to a different page. The valid links are added to a linked list. After all links of a page are added to the linked list, the links themselves are navigated in search for further links thus forming a recursive procedure. Each time a link is navigated a variable is incremented. As soon as the maximum number of links is navigated the downloading procedure starts
10. Add the following data members to the Dialog class
    CHttpFile* m_pHttpFile;
    CHttpConnection* m_pHttpConnection;
    CInternetSession m_ISession;
11. Using the **m_ISession** member variable call **GetHttpConnection( )** to obtain a connection pointer in **m_pHttpConnection**. The entire linked list formed is now iterated using **CStringList::GetHeadPosition ( )** & **CStringList::GetNext( )**. Each time the link is retrieved and passed to a user-defined function which is capable of downloading a file.
12. The user-defined function works by parsing the URL once again by calling **AfxParseURLEx( )** separating the object requested in the URL. Next this object is passed as an argument to **CHttpConnection::OpenRequest ( )**. **HttpConnection::SendRequest( )** sends the request to the server and obtains a pointer to the remote file in **CHttpFile**.
13. This Http File pointer along with additional data is passed to a worker thread responsible for downloading the file as it may take time. The worker thread uses the **CHttpFile::Read( )** the **CFile::Write( )** to make a local copy of the remote file.
14. Meanwhile the primary thread uses the **CHttpFile** pointer to call **CHttpFile::QueryInfo( )** which provides information about the length, status, etc. about the file begin downloaded and inserts it into the list control using **CListCtrl::InsertItem( )** and **CListCtrl::SetItemText( )**.
15. To synchronize main thread with the worker thread the function **CEvent::Lock( )** is used. This prevents the main thread from terminating before worker thread completes downloading of other files.

Further Improvements

1. The application should have its own browser.
2. Make a provision to allow users to display '.asp' files offline. For this the user would be required to create FTP directory in PWS or IIS. Refer the help file of these products.
3. Display progress bar when files are being downloaded.
4. Add filters so that only files with the specified extension should get downloaded. For example:
   a) only .htm files
   b) .htm and image files
   c) .htm, image and sound files
5. Add code to create target directory if not already in existence.
6. Add Complete Site download option. When this option is selected disable Maximum Links option.

Download